MRI Image Segmentation Project Report

Ruobing Bai, Sara Benedetti, Yakun Chen, Chun-Li Chuang,

Wanchen Geng, Ruiwen Jin,, Rohit Thakur, Zheying Yu

ABSTRACT

It has always been a challenge for brain surgeons to pinpoint the ventricle area within a patient's head. Surgeons so far have been operating by experience or educated guess when trying to locate it. To maximize the accuracy in locating the ventricle area for operation, we introduce machine learning. In this report, we discuss how we utilized the U-net structure convolution neural network to segment the ventricle area from a 3-D MRI scan. Throughout the training process, data augmentation is also utilized to resolve the shortage of samples, including downsampling and lateral flips. Our model consists of 1.4 million parameters and numerous convolution, max pooling and upsampling layers, and our convolution neural network provides a high prediction accuracy of the ventricle area from an input of an MRI scan.

1 INTRODUCTION

Minimizing the loss of misdiagnosis is always a hot topic for healthcare organizations. By gathering, storing and extracting the key values out of a mess of raw data, data analysis plays a key role of distributing and concluding the biological and medical information to healthcare organizations. A good example for this disciplinary application is the data analysis for the study of Alzheimer's Disease. The early stage of the disease includes memory loss, difficulty of thinking and concentrating, and then the patients will not be able to pursue regular daily tasks. Based on medical observation, the patients with Alzheimer's disease will be seen with a shrinkage in the brain and the ventricles are noticeably enlarged. With 3D scanning images, the structure of a patient's brain could be visualized to the doctors. However, the ventricles are not able to scan separately. Based on Rohini Paul Joseph, C. Senthil Singh, and M.Manikandan's article *Brain Tumor MRI Image Segmentation and Detection in Image Processing,* traditional way to get the structure of ventricles is using 3D image processing software and manually separate the ventricles from the whole brain image. However, since manual image processing depends on how well the physician handles the image, developing an accurate algorithm model that could automatically segmented ventricles from the brain image is needed. Nowadays, MRI image scan has become more popular than CT scan. Based on the MedicineNet Website, MRI scans use "powerful magnetic fields and radio frequency pulses to produce detailed pictures of organs and other internal body structures". Thus, it gives more detailed information about the inner organ than CTs.

However, since it gives more information about the inner structure, image processing using MRIs data is more difficult to get higher accuracy compared with image processing using CTs data. According to M. A. Balafar, A. R. Ramli, M. I. Saripan and S. Mashohor's article *Review of brain MRI image segmentation methods*, image segmentation around MRI datasets usually faces challenges to "improve the accuracy, precision and speed of segmentation methods". What's more, as many researchers are detecting tumors or other abnormal tissues or organs, the research about 3D image segmentation with a whole brain image as input and a small hollow cavity which also belongs to the brain as output is rare.

This project is concentrating on 3D MRI image segmentation. By developing a three dimensional convolution neural network model for data segmentation, the ventricle in each 3D MRI image could be separated out and help the further surgery planning. The project contains five parts: Data collection, data augmentation, building and training with three dimensional convolution neural network, evaluation of the model and the conclusion.

2 APPLICATION

From the application perspective, this project is to build CNN models to segment the 3D MRI images automatically, so that external ventricular drain (EVD) can be more easily performed. An EVD is a neurosurgery to treat hydrocephalus and relieve elevated intracranial pressure when the normal flow of cerebrospinal fluid (CSF) inside the brain is obstructed. In particular, the purpose of this surgery is to place a flexible plastic catheter which reduces the intracranial pressure by distracting CSF from the ventricles. Therefore, it is extremely important to partition the brain images into meaningful segments which only contain the ventricles, so that the plans for EVD can be made more precisely and more efficiently. By making an automatic MRI segmenter, this project is beneficial for enhancing the operability and accuracy of EVD.

3 DATA COLLECTION

The project is using a set of brain imaging data which consists of a collection of brain images from adults between 18 to 96 years old with both genders from OASIS database. Each subject includes 3 individual T1-weighted MRI scanned images. Among the individuals there are 100 people having Alzheimer's Disease (AD). 3D MRI images of ventricles are manually separated out by software 3D Slicer and converted to a python ready format. The project is based on 113 3D ventricles images which are sliced manually and changed the diagonals to [-1,-1,1,1] for preparation of future transformation. The project expects to have 3D ventricles image as output values and 3D whole MRI brain image as input values. Since the dataset is not large enough, data augmentation for both inputs and outputs are needed to prevent overfitting.

4 DATA AUGMENTATION

After loading 3D brain image and their segmentation data into Python Jupyter, the 3D image data is converted to a two dimensional matrix for future training and better understanding. By reshaping both ventricles and whole brain data with dimensions (87, 176, 208, 176), the sparse datasets have been sliced into 87 brain image slices, each with the size of image (176, 208, 176). For improvement of the model, since the segmented ventricles are small parts in the upper middle of the whole brain, the data values near the boundary of the 3D structure is useless and could be considered as noise data. Using the if statement to find the boundary of every ventricle, the image slices are cut into smaller cuboids with the size of (80, 120, 120). Because the sample size is not large, we used two methods to expand the size: downsampling and data augmentation.

4.1 DOWNSAMPLING

We attempted to separate every 3D brain image into 8 equal parts, so that the dimension of it was cut down from (176, 208, 176) to (88, 104, 88). However, the ventricle only accounts for a small proportion of the whole brain. So, some of the downsampled data do not contain the segmented parts. The consequence is that the output data are all zeros without any target number. As a result, most of the downsampled data are useless.

4.2 DATA AUGMENTATION TECHNIQUES

Since the sample size is not enough, in order to enlarge the dataset for the network in case of overfitting, other techniques of data augmentation have been chosen and tested. We made an effort to use the scipy package in python to slightly enlarge the 3D brain images. However, our available resources kept crashing during the process of enlarging the images. A possible reason for this is that the spicy.zoom() function is written basically for 2D images. Too many 3D images make the time complexity much higher. We then attempted to enlarge only some of the images. However, because the image is larger than before, there is not an efficient method to cut down the image as for the original dimension (176, 208, 176).

5 MODEL ARCHITECTURE

With the segmentation goal in mind, we found the U-Net architecture for CNN to be the most appropriate method to implement. It was developed by Olaf Ronneberger et al. for biomedical image segmentation in a research paper in 2015. The U-Net architecture can localize the borders and areas by classifying every pixel and maintains the same dimensionality as what the input provides. The model is structured from multiple convolutions and pooling layers, and it does not have any dense layer. The visualization of the architecture of the model is presented in *Appendix 1*. It consists of two paths, the contracting path, and the expansive path. The two paths are symmetric to each other, hence the origin of its name.

The contracting path follows the normal CNN process. In this project's instance, the samples have a dimension of 80 X 120 X 120, but the input shape needs to account for the feature channels within the upcoming convolution layers, hence the 4-dimensional input shape. Within each convolution layer, the 3-dimensional filter applies padded convolution and outputs a value for each voxel. After two convolution layers, 3D max-pooling is applied to take the largest element from the rectified feature map within a specified cuboid. We followed the standard structure of the original paper and applied max-pooling with a 2 X 2 X 2 cube, which cuts the input dimension by half after the operation. The downsampling steps are then repeated multiple times with each convolution layer having double the amount of feature channels as the previous repetition. This reduced the input dimension down to 10 by 15 by 15. This is then accompanied by two additional convoluted layers without max pooling. We then enter the expansive path.

In the expansive path, we're bringing the image back up to the original input size. It begins by upsampling the output of the previous layer. Here, we upsampled by the same size of what was used in the max-pooling layers earlier, which simply doubled each voxel to double up the size of each dimension. After upsampling, the 3D-image is concatenated with the corresponding 3D-image from the contracting path to combine the information from the previous layers to get a more precise prediction. This causes the feature channels to double again from the concatenation. However, it is then followed by two convolution layers with half of what the previous convolution layer has. Similar to the contracting path, the upsampling steps are then repeated the same amount of times as what is done in the contracting path with concatenation of each respective 3D-image in the contracting path. Finally, a convolution layer with two feature channels followed by another layer with one feature channel to produce the final trained output.

6 LEARNING PROCESS

Before reaching our final model, we tried training and testing on small samples of data to make best use of available computing power for getting promising results. In our dataset position of ventricles is indicated by ‘1’ and ‘0’ otherwise. Our model accuracy depended upon how correctly we predicted each voxel.

In our first trial, we divided each axis of image by half and splitted into training and testing sets by extracting only the middle 50% of voxels for each image while using 80% data for training and 20% for testing. Here our input tensor shape was (88,104,88,1) and we were able to get overall 98% accuracy on the test set. But the main issue with segmentation is the sparsity of the data. For testing our predictions we took a sample from the original dataset and listed out actual 1’s and predicted 1’s on the same. Calculated accuracy for correctly predicting 1’s was found out to be 88%. Also for this trial kernel size was set to 3x3x3 along with learning rate as 0.0001 and batch size of 1. Problem with this smaller model is that the real world position of ventricles can be out of these strict cube boundaries thus in further models we wanted to increase our search volume of ventricles.

In the second trial,we tried to adapt a better strategy for train test split. Here we implemented 5 folds cross validation to generate 80% training and 20% testing indices. Also for this trial we used augmented data to avoid any overfitting. Size of input tensor in this trial was (80,120,120,1). Also we further sampled data into smaller cuboids so as to generate 261 samples. For this model we got an accuracy of 97.34% with kernel size of 3x3x3 and learning rate of 0.0001.

In this trial, we tend to go for an overshooting approach for reducing the loss by gradually reducing the learning rate.We set the high learning rate initially lr=0.001. Following are the steps we followed. Loaded the last saved model and ran epochs with the same parameters as last trials except learning rate. Save model weights at each epoch. Stop at a point where loss no longer decreases. Then reduced learning rate and followed the same steps again.

7 VISUALIZATION OF FIRST LAYER WEIGHTS

Visualizing the weights of the first convolution layer can give insights about how the network is learning. Moreover, comparing the weights before and after the training of the model can intuitively show that the training worked and the network learned something: colors in the picture representing the weights before the training should appear random and noisy, while after the training patterns should emerge.

In our study, the 3D nature of the MRI and, as a consequence, of the convolution layers, made more challenging the visualization of the weights. For a given 3-dimensional filter, we investigated two different solutions: representing its weights slicing the matrix in one of its three dimensions, and visualizing them in a 3-dimensional plot. Indeed, a series of slices of the 3-dimensional filter can clearly show patterns that are aligned with the main axes of the matrix itself; however, patterns that do not follow such alignment will be less apparent. On the other hand, a 3-dimensional plot solves this issue, but in the representation the voxels that are more hidden will be less visible, possibly making the interpretation more difficult. Therefore, we chose to combine together these two approaches.

For the visualization, the weights were first normalized, obtaining a range of values between 0 and 1. We then compared colormaps in order to see which was the most effective to discern patterns. In particular, we compared the diverging colormap "RdYlGn" (Matplotlib) to a gray sequential colormap. The gray colormap gives the wrong illusion that the white pixels/voxels are an empty filling and thus less relevant, especially in the 3D visualization. Therefore, we decided to use the diverging colormap, where red pixels/voxels correspond to lower weight values and green pixels/voxels to higher values. For the 3-dimensional plot, we decided to discretize the range of colors and we chose three intervals of values corresponding to three equally spaced ranges of weight values. As a reminder, the images analysed in this study do not contain colors.

Finally, the code allows the user to specify the size of the 3-dimensional filters and the layer to be considered for the visualization of the weights.

7.1 ANALYSIS OF THE FIRST-LAYER WEIGHTS

We visualized the weights of the first convolution layer of the trained network used for the segmentation, having 3-dimensional filters of size of 3x3x3. We also visualized the weights of an identical model, but before its training (see Figure Appendix 1a). The weights after the training do not appear random like those before the training; however, the size of the filter (3x3x3) is too small to be able to discern patterns in it.

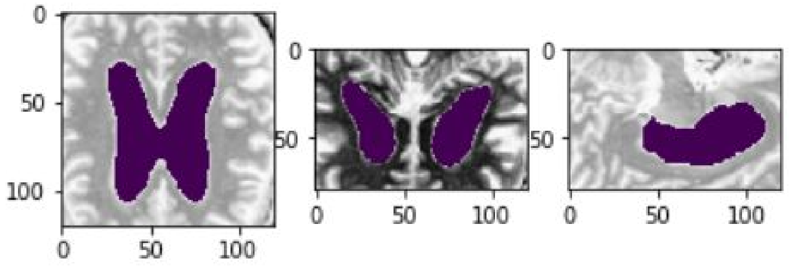
Therefore, we trained a network with an identical architecture except for the size of the filters of the first convolution layer, that was set to 7x7x7. The training was done for 28 epochs with the same training and validation data used for the previous model. The loss and validation loss were of about 0.38, and the accuracy and validation accuracy were of about 0.96. The visualization of the weights of the trained model, compared to the model before the training, did not show a significant difference: patterns did not seem to have emerged (see Appendix 1b). This might be an indicator that the training was not done for a sufficient number of epochs. We therefore tested if the network was successfully able to perform ventricular segmentation: the network seemed to have learned to classify everything, or almost, as the most common class and was still not performing correctly the segmentation, confirming our hypothesis. Therefore, the usefulness of the visualization tool for the weight was verified by this experiment. In future, it would be interesting to train the network again, for a longer time, in order to achieve a sufficient validation accuracy: in such a case, we expect that patterns will emerge.

8 RESULT

For our final model, we went back to the original sample image numbers. Also in U-net architecture we added two additional convolution layers, doubled learning rate to lr=0.0002. Further, we trained over this model and retrieved the epoch where loss was minimum. Then we performed a lateral shift over original data size to double the original sample size. Furthermore, reloaded the saved model and trained it on the doubled inputs.

Given the base accuracy was 97% we got overall accuracy of 99.01% for this model. We tested this model on completely new 2 samples and got a prediction accuracy of 80.58% , 31.79% respectively. Here accuracy depends upon size of the ventricle as well. The second sample was considerably harder to identify than the first one. But we were able to see some considerable results. The actual and predicted images are shown in Figure 1.

Actual Image Sample



Predicted Image Sample

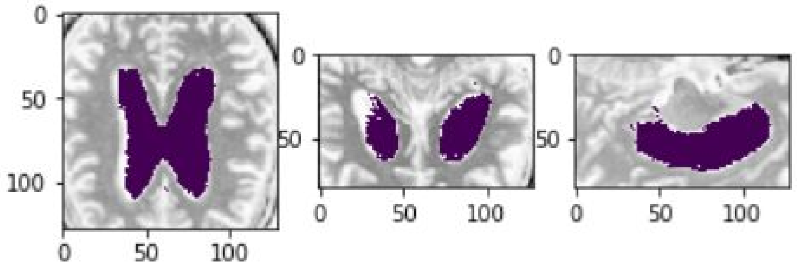


Figure 1: ActualVentricles Position and Predicted Ventricle Position

9 IMPROVEMENTS/FUTURE WORK

We identified a variety of improvements that can possibly be made to further improve model reliability. One out of which is to increase the number of data samples along with increasing computational power by making use of cloud computing resources such as AWS EC2 , Google Cloud Platform.

Another improvement can be made is to fine tune our model on training data. This includes use of the current model to predict training data and retrain it for wrongly predicted images using the weights of the previous model.

One more strategy for improving model accuracy further is to change current loss function into weighted binary cross entropy where a heavy penalty can be added for guessing 0's based on proportion of 0's and 1's in input data.

Future work of this project will focus on segmentation of multiple features from the brain's MRI scans by further changing model architecture into adding more convolution layers.

REFERENCE

1.“Alzheimer's Disease.” *Mayo Clinic*, Mayo Foundation for Medical Education and Research, 8 Dec. 2018, www.mayoclinic.org/diseases-conditions/alzheimers-disease/symptoms-causes/syc-20350447.

2. “Brain with Alzheimer's Disease.” *BrightFocus Foundation*, 9 Apr. 2019, www.brightfocus.org/alzheimers/infographic/brain-alzheimers-disease.

3. "CT Scan Vs. MRI Differences Between Safety, Cost, And Uses". *Medicinenet*, 2020, https://www.medicinenet.com/ct_scan_vs_mri/article.htm.

4. C., Senthil Singh, et al. “Brain Tumor Mri Image Segmentation And Detection In Image Processing.” International Journal of Research in Engineering and Technology, vol. 03, no. 13, 2014, pp. 1–5., doi:10.15623/ijret.2014.0313001.

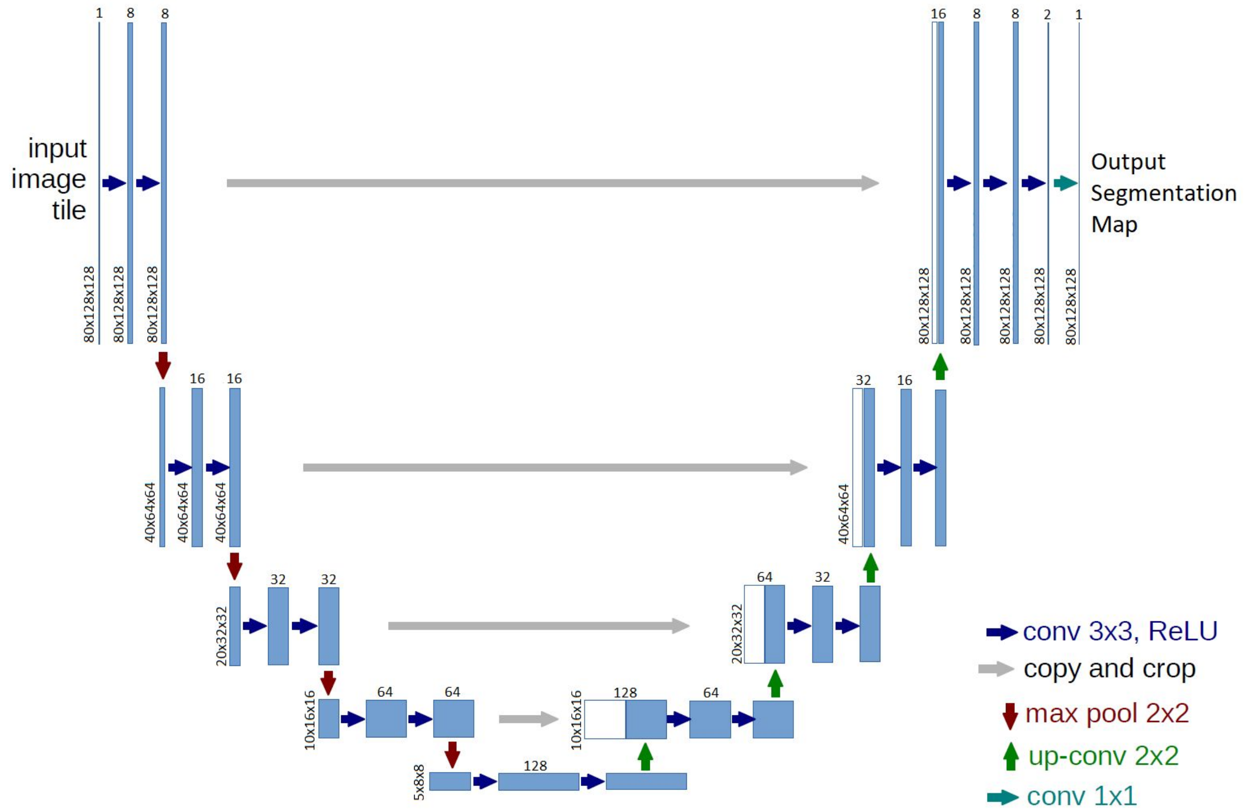
5. Balafar, M. A., et al. “Review of Brain MRI Image Segmentation Methods.” *Artificial Intelligence Review*, vol. 33, no. 3, 2010, pp. 261–274., doi:10.1007/s10462-010-9155-0.

6. Zhang, Jeremy. “UNet Line by Line Explanation.” Medium, Towards Data Science, 18 Oct. 2019, towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5.

7. Ronneberger, Olaf, et al. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015, https://arxiv.org/pdf/1505.04597.pdf.

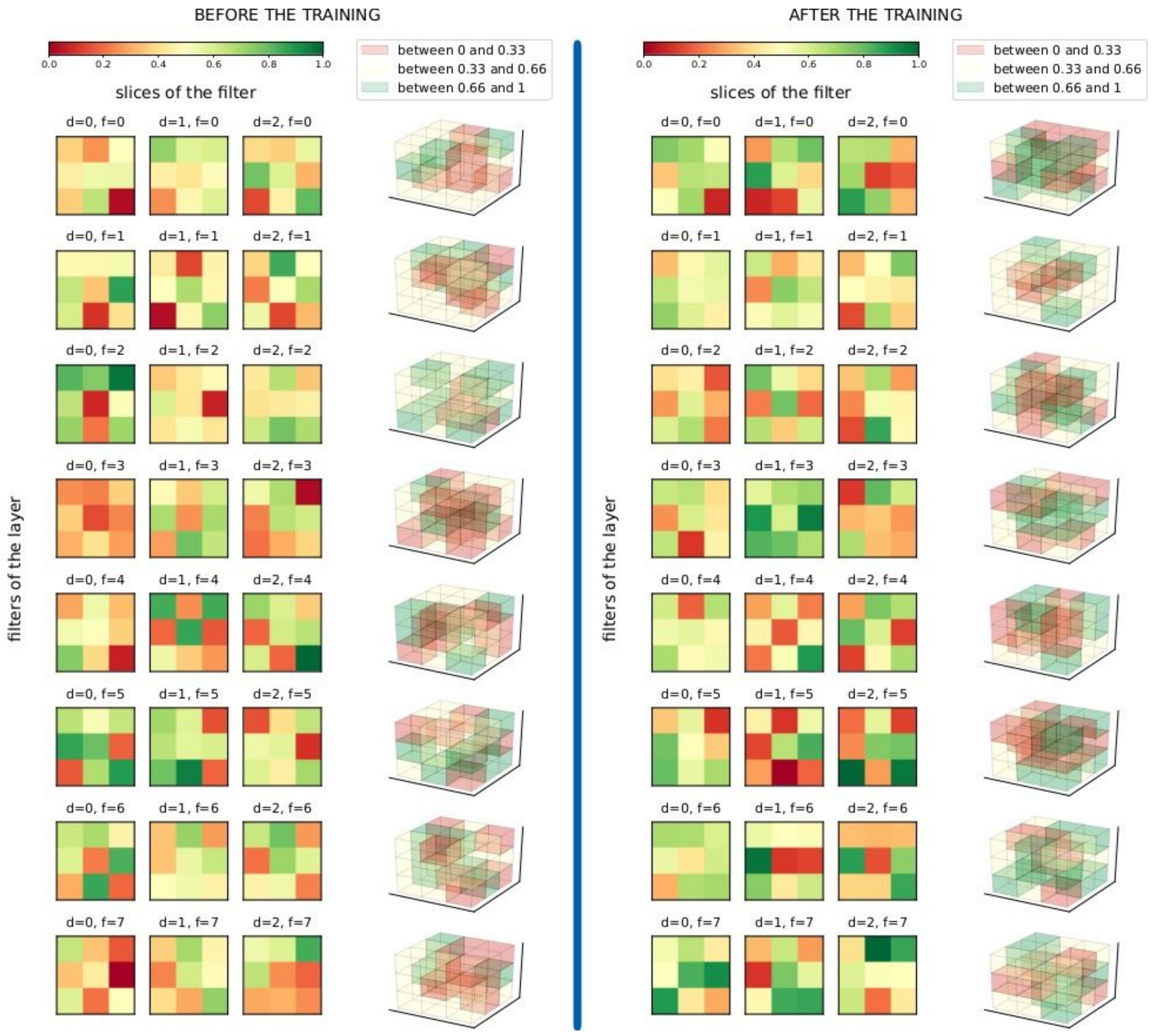
APPENDIX

Appendix 1

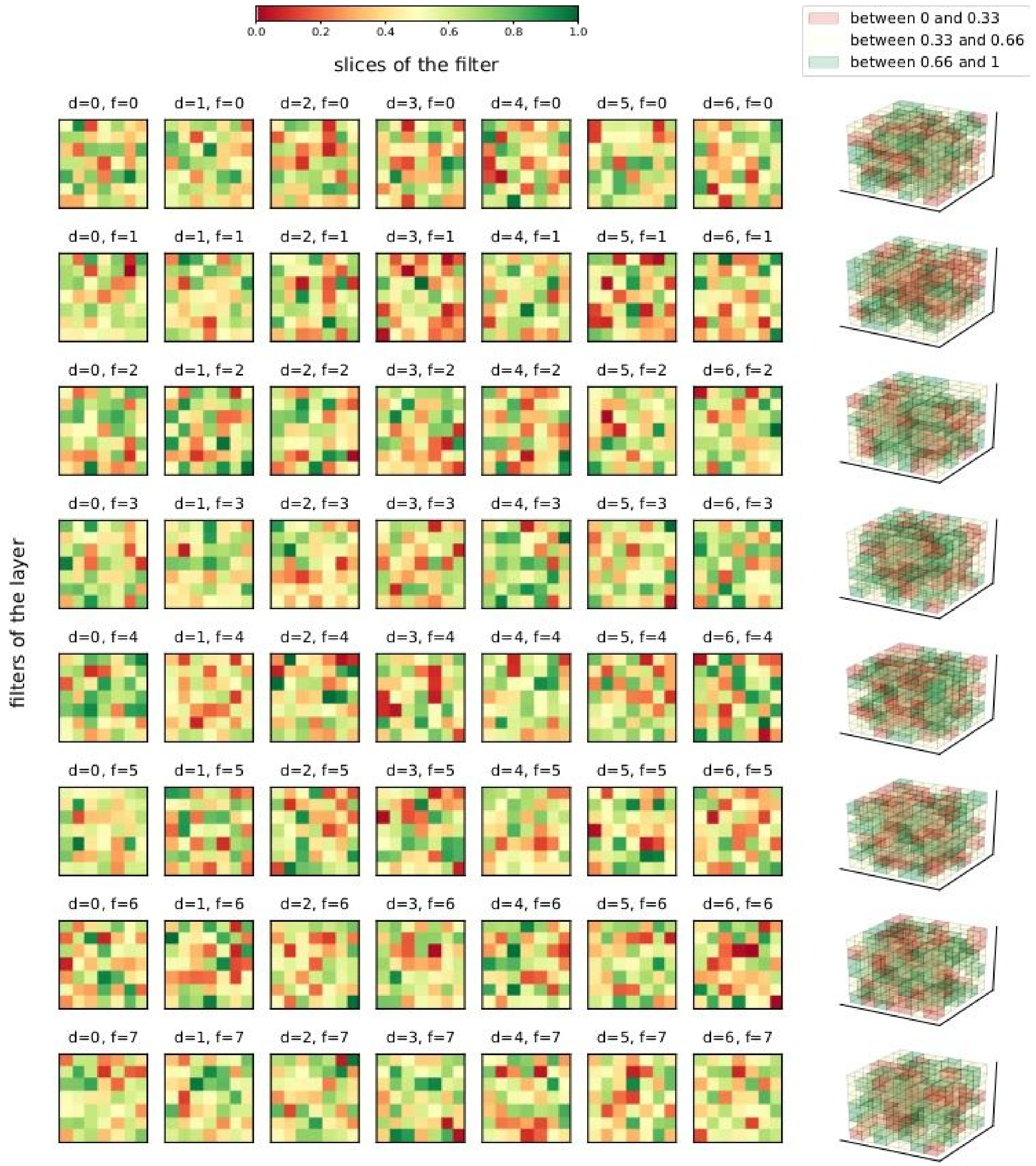


Appendix 1: The U-Net Model Architecture Visualization of the model used in the Result section.

Appendix 2



Appendix 2a: Visualization of the weights of the first convolution layer (filter size: 3x3x3) of the model used for the segmentation, before and after the training; f: filter number, d: slice number (d stands for depth). The weights are normalized, in order to have a range of values between 0 (dark red) and 1 (dark green).



Appendix 2b: Visualization of the weights of the first convolution layer (filter size of 7x7x7) of the model trained for 28 epochs; f: filter number, d: slice number (d stands for depth). The weights are normalized, in order to have a range of values between 0 (dark red) and 1 (dark green).